



HAL
open science

Higher-order Clustering and Pooling for Graph Neural Networks

Alexandre Duval, Fragkiskos D. Malliaros

► **To cite this version:**

Alexandre Duval, Fragkiskos D. Malliaros. Higher-order Clustering and Pooling for Graph Neural Networks. CIKM 2022 - 31st ACM International Conference on Information & Knowledge Management, Oct 2022, Atlanta, Georgia, United States. pp.426-435, 10.1145/3511808.3557353 . hal-03910823

HAL Id: hal-03910823

<https://inria.hal.science/hal-03910823>

Submitted on 22 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Higher-order Clustering and Pooling for Graph Neural Networks

Alexandre Duval

Université Paris-Saclay, CentraleSupélec, Inria
Gif-sur-Yvette, France

alexandre.duval@centralesupelec.fr

Fragkiskos D. Malliaros

Université Paris-Saclay, CentraleSupélec, Inria
Gif-sur-Yvette, France

fragkiskos.malliaros@centralesupelec.fr

ABSTRACT

Graph Neural Networks achieve state-of-the-art performance on a plethora of graph classification tasks, especially due to pooling operators, which aggregate learned node embeddings hierarchically into a final graph representation. However, they are not only questioned by recent work showing on par performance with random pooling, but also ignore completely higher-order connectivity patterns. To tackle this issue, we propose HoscPOOL, a clustering-based graph pooling operator that captures higher-order information hierarchically, leading to richer graph representations. In fact, we learn a probabilistic cluster assignment matrix end-to-end by minimising relaxed formulations of motif spectral clustering in our objective function, and we then extend it to a pooling operator. We evaluate HoscPOOL on graph classification tasks and its clustering component on graphs with ground-truth community structure, achieving best performance. Lastly, we provide a deep empirical analysis of pooling operators' inner functioning. The code is available here.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Computing methodologies** → **Machine learning algorithms**.

KEYWORDS

Graph Neural Networks (GNNs), Graph Pooling, Clustering.

1 INTRODUCTION

Graph Neural Networks are powerful tools for graph datasets due to their message passing scheme, where they propagate node features along the edges of the graph to compute meaningful node representations [14, 17]. They achieve state-of-the-art performance on a variety of tasks including clustering, link prediction, node and graph classification [53]. For the latter, since the goal is to predict the label of the entire graph, standard approaches pool together all nodes' embeddings to create a single graph representation, usually via a simple sum or average operation [2]. This *global pooling* discards completely graph structure when computing its final representation, failing to capture the topology of many real-world networks and thus preventing researchers to build effective GNNs.

More desirable alternatives emerged to solve this limitation. They progressively coarsen the graph between message passing layers,

for instance by regrouping highly connected nodes (i.e. clusters) together into supernodes with adapted adjacency / feature vectors. This allows to better capture the graph hierarchical structure compared to global pooling, without losing relevant information if the coarsening is accurately done. While the first clustering-based pooling algorithms were deterministic [9, 12] – because of their high computational complexity, their transductive nature and their incapacity to leverage node features – they were replaced by trainable *end-to-end clustering* approaches such as STRUCTPOOL [51] or DIFFPOOL [49]. Such methods solve the above limitations, often by learning a cluster assignment matrix along with GNN parameters thanks to a specific loss function, e.g. a link prediction score.

Despite presenting many advantages, such methods pool nodes together based on a simple functions or metrics which often lack strong supporting theoretical foundations. Besides, they reduce the graph uniquely based on first-order information. And in many cases, graph datasets may not present any edge-based connectivity structure, leading to insignificant graph coarsening steps, while they may have clear community structure with respect to more complex (domain-specific) motifs [21]. Overall, this limits the expressiveness of the hierarchical information captured, and therefore of the classification performance. On top of that, existing pooling operators were surprisingly shown to perform on par with random pooling for many graph classification tasks, raising major concerns [27] and finding limited justifications. This discovery appears rather counter-intuitive as we logically expect the graph coarsening step, that is, the way to pool nodes together, to increase significantly the graph hierarchical information captured in its final representation.

Combining these two facts, we propose HoscPOOL, a new end-to-end higher-order pooling operator grounded on probabilistic motif spectral clustering to capture a more advanced type of communities thanks to the incorporation of higher-order connectivity patterns. The latter has shown to be very successful for a wide range of applications [20] but has not yet been applied to graph classification, while it could greatly benefit from it. Specifically, we hierarchically coarsen the input graph using a cluster assignment matrix S learned by defining a well-motivated objective function, which includes continuous relaxations of motif conductance and thus combines various types of connectivity patterns for greater expressiveness. Since the process is fully differentiable, we can stack several such pooling layers, intertwined by message passing layers, to capture graph hierarchical information. We jointly optimise our unsupervised loss with any task-specific supervised loss function to allow truly end-to-end graph classification. Finally, we evaluate the performance of HoscPOOL on a plethora of graph datasets, and the reliability of its clustering algorithm on a variety of graphs endowed with ground-truth community structure. During this experiment phase, we proceed to a deep analysis aimed to understand

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9236-5/22/10...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

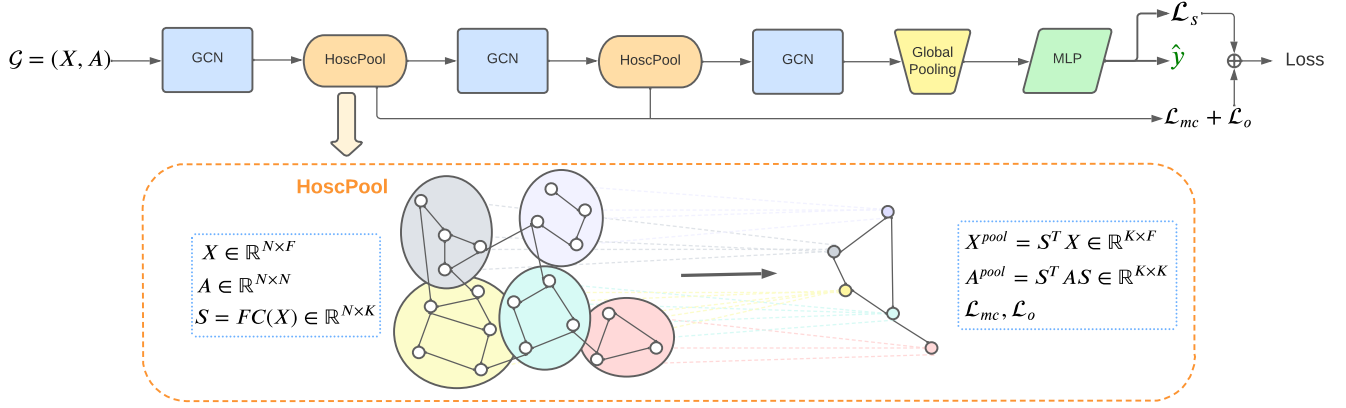


Figure 1: A graph classification pipeline with HoscPool hierarchical pooling to reduce graph \mathcal{G} to $\mathcal{G}_{pool} = (X_{pool}, A_{pool})$ via a cluster assignment matrix S learned end-to-end from a motif spectral clustering inspired loss function $\mathcal{L}_{mc} + \mathcal{L}_o$.

why existing pooling methods fail to truly outperform random baselines and attempt to provide explanations. This is another important contribution, which we hope will help future works.

2 RELATED WORK

Graph pooling. Leaving aside global pooling [2, 39, 47], we distinguish between two main types of hierarchical approaches. Node drop methods [3, 13, 19, 31, 48, 50, 52] use a learnable scoring function based on message passing representations to assess all nodes and drop the ones with lowest score. The drawback is that we lose information during pooling by dropping completely certain nodes. On the other hand, clustering approaches cast the pooling problem as a clustering one [10, 23–25, 33, 46, 51]. For instance, STRUCTPOOL [51] utilizes conditional random fields to learn the cluster assignment matrix; HAARPOOL [46] uses the compressive Haar transform; EDGEPOOL [10] gradually merges nodes by contracting high-scoring edges. Of particular interest here are two very popular end-to-end clustering methods, namely DIFFPOOL [49] and MINCUTPOOL [5], because of their original and efficient underlying idea. While DIFFPOOL utilises a link prediction objective along with an entropy regularization to learn the cluster assignment matrix, MINCUTPOOL leverages a min-cut score objective along with an orthogonality term. Although there are more pooling operators, we wish to improve this line of method, that we think is promising and perfectible. In addition to solving existing limitations, we want to introduce the notion of higher-order to pooling for graph classification, which is unexplored yet.

Higher-order connectivity patterns (i.e. motifs – small network subgraphs like triangles ∇), are known to be the fundamental building blocks of complex networks [6, 28]. They are essential for modelling and understanding the organization of various types of networks. For instance, they play an essential role in the characterisation of social, biological or molecules networks [30]. [11] showed that vertices participating in the same higher-order structure often share the same label, spreading its adoption to node classification tasks [20, 22]. Going further, several recent research papers have

clearly demonstrated the benefits of leveraging higher-order structure for link prediction [1, 37], explanation generation [32, 36], ranking [34], clustering [15, 18]. Regarding the latter, [4, 42] argue that domain-specific motifs are a better signature of the community structure than simple edges. Their intuition is that motifs allow us to focus on particular network substructures that are important for networks of a given domain. As a result, they generalized the notion of conductance to triangle conductance (Section 3), which was found highly beneficial by [6, 40].

3 PRELIMINARY KNOWLEDGE

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph with vertex set \mathcal{V} and edge set \mathcal{E} , characterised by its adjacency matrix $A \in \mathbb{R}^{N \times N}$ and node feature matrix $X \in \mathbb{R}^{N \times F}$. $D = \text{diag}(A \mathbf{1}_N)$ is the degree matrix and $L = D - A$ the Laplacian matrix of \mathcal{G} . $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$ is the symmetrically normalised adjacency matrix with corresponding \tilde{D}, \tilde{L} .

3.1 Graph Cut and Normalised Cut

Clustering involves partitioning the vertices of a graph into K disjoint subsets with more intra-connections than inter-connections [43]. One of the most common and effective way to do it [35] is to solve the Normalised Cut problem [38]:

$$\min_{S_1, \dots, S_K} \sum_{k=1}^K \frac{\text{cut}(S_k, \bar{S}_k)}{\text{vol}(S_k)}, \quad (1)$$

where $\bar{S}_k = \mathcal{V} \setminus S_k$, $\text{cut}(S_k, \bar{S}_k) = \sum_{i \in S_k, j \in \bar{S}_k} A_{ij}$, and $\text{vol}(S_k) = \sum_{i \in S_k, j \in \mathcal{V}} A_{ij}$. Unlike the simple min-cut objective, (1) scales each term by the cluster volume, thus enforcing clusters to be “reasonably large” and avoiding degenerate solutions where most nodes are assigned to a single cluster. Although minimising (1) is NP-hard [44], there are approximation algorithms with theoretical guarantees [8] for finding clusters with small conductance, such as Spectral Clustering (SC), which proposes clusters determined based on the eigen-decomposition of the Laplacian matrix. A refresher on SC is provided in [43].

3.2 Motif conductance

While the Normalised Cut builds on first-order connectivity patterns (i.e. edges), [4, 42] propose to cluster a network based on specific higher-order substructures. Formally, for graph \mathcal{G} , motif M made of $|M|$ nodes, and $\mathcal{M} = \{\mathbf{v} \in \mathcal{V}^{|M|} | \mathbf{v} = M\}$ the set of all instances of M in \mathcal{G} , they propose to search for the partition $\mathcal{S}_1, \dots, \mathcal{S}_K$ minimising motif conductance:

$$\min_{\mathcal{S}_1, \dots, \mathcal{S}_K} \sum_{k=1}^K \frac{\text{cut}_M^{(\mathcal{G})}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\text{vol}_M^{(\mathcal{G})}(\mathcal{S}_k)}, \quad (2)$$

where $\text{cut}_M^{(\mathcal{G})}(\mathcal{S}_k, \bar{\mathcal{S}}_k) = \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}(\exists i, j \in \mathbf{v} | i \in \mathcal{S}_k, j \in \bar{\mathcal{S}}_k)$, i.e. the number of instances \mathbf{v} of M with at least one node in \mathcal{S}_k and at least one node in $\bar{\mathcal{S}}_k$; and $\text{vol}_M^{(\mathcal{G})}(\mathcal{S}_k) = \sum_{\mathbf{v} \in \mathcal{M}} \sum_{i \in \mathbf{v}} \mathbf{1}(i \in \mathcal{S}_k)$, i.e. the number of motif instance endpoints in \mathcal{S}_k .

4 PROPOSED METHOD

The objective of this paper is to design a differentiable cluster assignment matrix \mathbf{S} that learns to find relevant clusters based on higher-order connectivity patterns, in an end-to-end manner within any GNN architecture. To achieve this, we formulate a continuous relaxation of motif spectral clustering and embed the derived formulation into the model objective function to enforce its learning.

4.1 Probabilistic motif spectral clustering

Before exploring how we can rewrite the motif conductance optimisation problem (2) in a solvable way, we introduce the motif adjacency matrix A_M , where each entry $(A_M)_{ij}$ represents the number of motifs in which both node i and node j participate. Its diagonal has zero values. Formally, $(A_M)_{ij} = \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}(i, j \in \mathbf{v}, i \neq j)$. \mathcal{G}_M is the graph induced by A_M . $(D_M)_{ii} = \sum_{j=1}^N (A_M)_{ij}$ and L_M are the motif degree and motif Laplacian matrices.

For now, we focus on triangle motifs ($M = K_3$), and extend to more complex motifs in Section 4.2. From [4], we have:

$$\begin{aligned} \text{cut}_M^{(\mathcal{G})}(\mathcal{S}_k, \bar{\mathcal{S}}_k) &= \frac{1}{2} \sum_{i \in \mathcal{S}_k} \sum_{j \in \bar{\mathcal{S}}_k} (A_M)_{ij} \\ \text{vol}_M^{(\mathcal{G})}(\mathcal{S}_k) &= \frac{1}{2} \sum_{i \in \mathcal{S}_k} \sum_{j \in \mathcal{V}} (A_M)_{ij}, \end{aligned}$$

which enables us to rewrite (2) as:

$$\begin{aligned} \min_{\mathcal{S}_1, \dots, \mathcal{S}_K} \sum_{k=1}^K \frac{\sum_{i \in \mathcal{S}_k, j \in \bar{\mathcal{S}}_k} (A_M)_{ij}}{\sum_{i \in \mathcal{S}_k, j \in \mathcal{V}} (A_M)_{ij}} \\ \equiv \max_{\mathcal{S}_1, \dots, \mathcal{S}_K} \sum_{k=1}^K \frac{\sum_{i, j \in \mathcal{S}_k} (A_M)_{ij}}{\sum_{i \in \mathcal{S}_k, j \in \mathcal{V}} (A_M)_{ij}}, \end{aligned} \quad (3)$$

where the last equivalence follows from

$$\sum_{i, j \in \mathcal{S}_k} (A_M)_{ij} + \sum_{i \in \mathcal{S}_k, j \in \bar{\mathcal{S}}_k} (A_M)_{ij} = \sum_{i \in \mathcal{S}_k, j \in \mathcal{V}} (A_M)_{ij}.$$

Instead of using partition sets, we define a discrete cluster assignment matrix $\mathbf{S} \in \{0, 1\}^{N \times K}$ where $S_{ij} = 1$ if $v_i \in \mathcal{S}_j$ and 0 otherwise. We denote by $\mathbf{S}_j = [S_{1j}, \dots, S_{Nj}]^\top$ the j^{th} column of \mathbf{S} ,

which indicates the nodes belonging to cluster \mathcal{S}_j . Using this, we transform (3) into:

$$\begin{aligned} \max_{\mathbf{S} \in \{0, 1\}^{N \times K}} \sum_{k=1}^K \frac{\sum_{i, j \in \mathcal{V}} (A_M)_{ij} S_{ik} S_{jk}}{\sum_{i, j \in \mathcal{V}} S_{ik} (A_M)_{ij}} \\ \equiv \max_{\mathbf{S} \in \{0, 1\}^{N \times K}} \sum_{k=1}^K \frac{\mathbf{S}_k^\top A_M \mathbf{S}_k}{\mathbf{S}_k^\top D_M \mathbf{S}_k} \\ \equiv \min_{\mathbf{S} \in \{0, 1\}^{N \times K}} -\text{Tr} \left(\frac{\mathbf{S}^\top A_M \mathbf{S}}{\mathbf{S}^\top D_M \mathbf{S}} \right), \end{aligned} \quad (4)$$

where the division sign in the last line is an element-wise division on the diagonal of both matrices. By definition, \mathbf{S} is subject to the constraint $\mathbf{S} \mathbf{1}_K = \mathbf{1}_N$, i.e. each node belongs exactly to 1 cluster.

This optimisation problem is NP-hard since \mathbf{S} take discrete values. We thus relax it to a probabilistic framework, where \mathbf{S} take continuous values in the range $[0, 1]$, representing cluster membership probabilities, i.e. each entry S_{ik} denotes the probability that node i belongs to cluster k . Referring to [43] and [4], solving this continuous relaxation of motif spectral clustering approximates a closed form solution with theoretical guarantees, provided by the Cheeger inequality [8]. Compared to the original hard assignment problem, this soft cluster assignment formulation is less likely to be trapped in local minima [16]. It also allows to generalise easily to multi-class assignment, expresses uncertainty in clustering, and can be optimised within any GNN.

4.2 End-to-end clustering framework

In this section, we leverage this probabilistic approximation of motif conductance to learn our cluster assignment matrix \mathbf{S} in a trainable manner. Our method addresses the limitations of (motif) spectral clustering: we cluster nodes based both on graph topology and node features; leverage higher-order connectivity patterns; avoid the expensive eigen-decomposition of the motif Laplacian; and allow to cluster out-of-sample graphs.

We compute the soft cluster assignment matrix \mathbf{S} using one (or more) fully connected layer(s), mapping each node's representation \mathbf{X}_{i*} to its probabilistic cluster assignment vector \mathbf{S}_{i*} . We apply a softmax activation function to enforce the constraint inherited from (4): $S_{ij} \in [0, 1]$ and $\mathbf{S} \mathbf{1}_K = \mathbf{1}_N$:

$$\mathbf{S} = \text{FC}(\mathbf{X}; \Theta). \quad (5)$$

Θ are trainable parameters, optimised by minimising the unsupervised loss function \mathcal{L}_{mc} , which approximates the relaxed formulation of the motif conductance problem (4):

$$\mathcal{L}_{mc} = -\frac{1}{K} \cdot \text{Tr} \left(\frac{\mathbf{S}^\top A_M \mathbf{S}}{\mathbf{S}^\top D_M \mathbf{S}} \right). \quad (6)$$

Referring to the spectral clustering formulation¹, $\mathcal{L}_{mc} \in [-1, 0]$. It reaches -1 when \mathcal{G}_M has $\geq K$ connected components (no motif endpoints are separated by clustering), and 0 when for each pair of nodes participating in the same motif (i.e. $(A_M)_{ij} > 0$), the cluster assignments are orthogonal: $\langle \mathbf{S}_{i*}, \mathbf{S}_{j*} \rangle = 0$. \mathcal{L}_{mc} is a non-convex function and its minimisation can lead to local minima, although

¹The largest eigenvalue $\mathbf{A}_M \mathbf{S} = \lambda \mathbf{D}_M \mathbf{S}$ is 1 and the smallest 0; we are summing only the k largest eigenvalues.

our probabilistic membership formulation makes it less likely to happen w.r.t. hard membership [16].

In fact, we allow the *combination of several motifs* inside our objective function (6) via $\mathcal{L}_{mc} = \sum_j \alpha_j \mathcal{L}_{mc^j}$ where \mathcal{L}_{mc^j} denotes the objective function with respect to a particular motif (e.g., edge \mathbf{I} , triangle ∇ , 4-nodes cycle \square) and α_j is an importance factor. This also increases the power of our method, allowing us to find communities of nodes w.r.t. a hierarchy of higher-order substructures. As a result, the graph coarsening step will pool together more relevant groups of nodes, potentially capturing more relevant patterns in subsequent layers, ultimately producing richer graph representation. We implement it for edge and triangle motifs:

$$\mathcal{L}_{mc} = -\frac{\alpha_1}{K} \cdot \text{Tr} \left(\frac{\mathbf{S}^\top \mathbf{A} \mathbf{S}}{\mathbf{S}^\top \mathbf{D} \mathbf{S}} \right) - \frac{\alpha_2}{K} \cdot \text{Tr} \left(\frac{\mathbf{S}^\top \mathbf{A}_M \mathbf{S}}{\mathbf{S}^\top \mathbf{D}_M \mathbf{S}} \right). \quad (7)$$

We let α_1, α_2 , to be dynamic functions of the epoch, subject to $\alpha_1 + \alpha_2 = 1$, allowing to first optimise higher-order motifs before moving on to smaller ones. It helps refine the level of granularity progressively and was found desirable empirically. This is the higher-order clustering formulation that we consider in the paper.

In case we would like to enforce more rigorously the hard cluster assignment, characteristic of the original motif conductance formulation, we design an auxiliary loss function:

$$\mathcal{L}_o = \frac{1}{\sqrt{K} - 1} \left(\sqrt{K} - \frac{1}{\sqrt{N}} \sum_{j=1}^K \|S_{*,j}\|_F \right), \quad (8)$$

where $\|\cdot\|_F$ indicates the Frobenius norm. This orthogonality loss encourages more balanced and discrete clusters (i.e. a node assigned to a cluster with high probability, while to other clusters with a low one), discouraging further degenerate solutions. Although its effect overlaps with \mathcal{L}_{mc} , it often smoothes out the optimisation process and even improves slightly performance in complex tasks or networks, such as graph classification. In (8), we rescale \mathcal{L}_o to $[0, 1]$, making it commensurable to \mathcal{L}_{mc} . As a result, the two terms can be safely summed and optimised together when specified. A parameter μ controls the strength of this regularisation.

Similarly to other cluster-based pooling operators, our method relies on two assumptions. Firstly, nodes are identifiable via their features. Secondly, node features represent a good initialisation for computing cluster assignments. The latter is realistic due to the homophily property of many real-world networks [26] as well as the smoothing effect of message passing layers [7], which render connected nodes more similar.

We conclude this section with a note for future work. An interesting research direction would be to extend this framework to 4-nodes motifs. Despite having managed to derive a theoretical formulation for the 4-nodes motif conductance problem in Appendix C, it becomes complex and would probably necessitate its own dedicated research, as it could be an promising extension.

4.3 Higher-order graph coarsening

The methodology detailed in the previous sections is a general clustering technique that can be used for any clustering tasks on any graph dataset. In this paper, we utilise it to form a pooling operator, called Hoscpool, which exploits the cluster assignment matrix \mathbf{S} to generate a coarsened version of the graph (with fewer

nodes and edges) that preserve critical information and embeds higher-order connectivity patterns. More precisely, it coarsens the existing graph by creating super-nodes from the derived clusters, with a new edge set and feature vector, depending on previous nodes belonging to this cluster. Mathematically,

$$\text{Hoscpool} : \mathcal{G} = (\mathbf{X}, \mathbf{A}) \rightarrow \mathcal{G}^{pool} = (\mathbf{X}^{pool}, \mathbf{A}^{pool}) \\ \mathbf{A}^{pool} = \mathbf{S}^\top \mathbf{A} \mathbf{S} \text{ and } \mathbf{X}^{pool} = \mathbf{S}^\top \mathbf{X}.$$

Each entry $X_{i,j}^{pool}$ denotes feature j 's value for cluster i , calculated as a sum of feature j 's value for the nodes belonging to cluster i , weighted by the corresponding cluster assignment scores. $\mathbf{A}^{pool} \in \mathbb{R}^{K \times K}$ is a symmetric matrix where $A_{i,j}^{pool}$ can be viewed as the connection strength between cluster i and cluster j . Given our optimisation function, it will be a diagonal-dominant matrix, which will hamper the propagation across adjacent nodes. For this reason, we remove self-loops. We also symmetrically normalise the new adjacency matrix. Lastly, note that we use the original \mathbf{A} and \mathbf{X} for this graph coarsening step; their motif counterparts \mathbf{A}_M and \mathbf{X}_M are simply leveraged to compute the loss function. Our work thus differ clearly from diffusion methods and traditional GNN leveraging higher-order.

Because our GNN-based implementation of motif spectral clustering is fully differentiable, we can stack several Hoscpool layers, intertwined with message passing layers, to hierarchically coarsen the graph representation. In the end, a global pooling and some dense layers produce a graph prediction. The parameters of each Hoscpool layer can be learned end-to-end by jointly optimizing:

$$\mathcal{L} = \mathcal{L}_{mc} + \mu \mathcal{L}_o + \mathcal{L}_s, \quad (9)$$

where \mathcal{L}_s denotes any supervised loss for a particular downstream task (here the cross entropy loss). This way, we should be able to hierarchically capture relevant graph higher-order structure while learning GNN parameters so as to ultimately better classify the graphs within our dataset.

4.4 Comparison with relevant baselines

Before moving to the experiments, we take a moment to emphasise the key differences with respect to core end-to-end clustering-based pooling baselines. We focus on MINCUTPOOL in the following since it is our closest baseline. DIFFPOOL and others differ more significantly, in addition to being less theoretically-grounded and efficient.

Firstly, MINCUTPOOL focuses on first-order connectivity patterns, while we work on higher-order, which implies a more elaborated background theory with the construction and combination of several motif adjacency matrices (each specific to a particular motif). This shall lead to capturing more advanced types of communities, producing ultimately a better coarsening of the graph. Secondly, we approximate a probabilistic version of the motif conductance problem (extension of the normalised min-cut to motifs) whereas MINCUTPOOL approximates the relaxed unnormalised min-cut problem. Despite claiming to formulate a relaxation of the normalised min-cut (a trace ratio), it truly minimises a ratio of traces in the objective function: $-\frac{\text{Tr}(\mathbf{S}^\top \tilde{\mathbf{A}} \mathbf{S})}{\text{Tr}(\mathbf{S}^\top \tilde{\mathbf{D}} \mathbf{S})}$. Since $\text{Tr}(\mathbf{S}^\top \tilde{\mathbf{D}} \mathbf{S}) = \sum_{i \in \mathcal{V}} \tilde{D}_{ii}$ is a constant, this yields the unnormalised min-cut $-\text{Tr}(\mathbf{S}^\top \tilde{\mathbf{A}} \mathbf{S})$, which often produces degenerate solutions. To cope with this limitation,

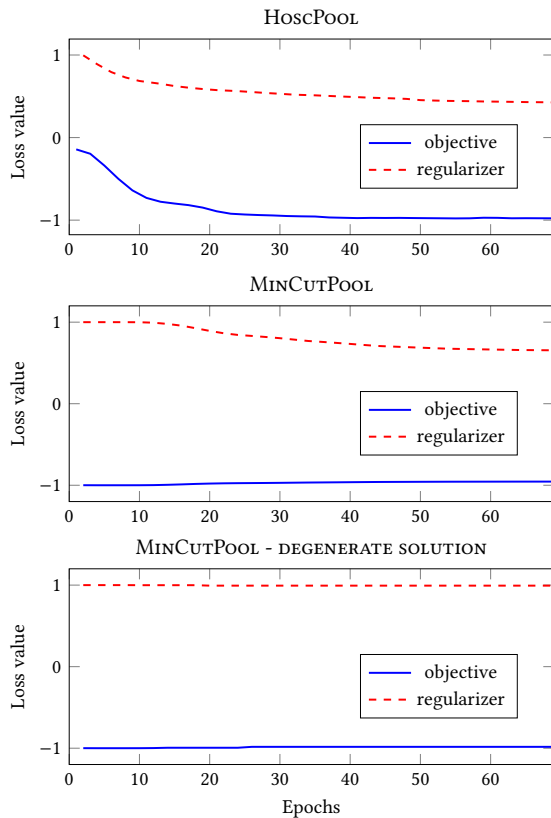


Figure 2: Loss function value w.r.t. epochs. MINCUTPOOL optimises the orthogonality loss, which decreases smoothly, while its min-cut objective remains constant (acting like a regularizer); whereas HosCPOOL optimises the main objective directly. Sometimes, MINCUTPOOL does not manage to optimise the regularizer loss, yielding a degenerate clustering.

MINCUTPOOL optimises in parallel a penalty term \mathcal{L}_o encouraging balanced and discrete clusters assignments. But despite this regularizer, it often gets stuck in local minima [45] (see Fig. 2), as we will see empirically in Section 5. We spot and correct this weakness in HosCPOOL. Thirdly, we introduced a new and more powerful orthogonality term together with a regularization control parameter. Unlike MINCUTPOOL, it is unnecessary but often smoothes out training and improves performance. Lastly, we showcase a different architecture involving a more general way of computing S .

5 EVALUATION

We now evaluate the benefits of the proposed method, with the goal of answering the following questions:

- (1) Does our differentiable higher-order clustering algorithm compute meaningful clusters? Is considering higher-order structures beneficial?
- (2) How does HosCPOOL compare with state-of-the-art pooling approaches for graph classification tasks?

- (3) Why do existing pooling operators fail to outperform significantly random pooling?

5.1 Clustering

Experimental setup. For this experiment, we first run a Message Passing (MP) layer; in this case a GCN model with skip connection for initial features [30]: $\tilde{X} = \text{ReLU}(AX\Theta_1 + X\Theta_2)$, where Θ_1 and Θ_2 are trainable weight matrices. It has 32 hidden units and ReLU activation function. We then run a Multi-Layer Perceptron (MLP) with 32 hidden units to produce the cluster assignment matrix of dimension $num_nodes \times num_clusters$, trained end-to-end by optimising the unsupervised loss function $\mathcal{L}_{mc} + \mu\mathcal{L}_o$. This architecture is trained using a learning rate of 0.001 for an Adam optimizer, 500 epochs, a gradient clip of 2.0, 200 early stop patience, a learning decay patience of 25 and $\mu = \{0, .1, 1\}$.

Metrics. We evaluate the quality of S by comparing the distribution of true node labels with the one of predicted labels, via Normalised Mutual Information $NMI(\tilde{y}, y) = \frac{H(\tilde{y}) - H(\tilde{y}|y)}{\sqrt{H(\tilde{y}) - H(y)}}$, where $H(\cdot)$ is the entropy and node cluster membership is determined by the argmax of its assignment probabilities. We also calculate completeness, modularity, normalised cut, and motif conductance (App. Table 7).

Datasets. We use a collection of node classification datasets with ground truth community labels: citation networks *Cora*, *PubMed*; collaboration networks *DBLP*, *Coauthor CS*; co-purchase networks *Amazon Photo*, *Amazon PC*; the *KarateClub* community network; and communication networks *Polblogs* and *Eu-email*. They are all taken from Pytorch Geometric. We construct three synthetic datasets: *Syn1*, *Syn2*, *Syn3* (based on several random graphs) where node labels are determined based on higher-order community structure and node features are simple graph statistics (Appendix A). They are designed to show the additional efficiency of HosCPOOL when datasets have clear higher-order structure, which is not always the case for the standard baseline datasets chosen.

Baselines. We compare HosCPOOL with the original spectral clustering (SC), motif spectral clustering (MSC)² as well as key pooling baselines DIFFPOOL and MINCUTPOOL. We refer to all methods by their pooling name for simplicity, although this experiment focuses on the clustering part and does not involve the coarsening step. We repeat all experiments 10 times and average results across runs. For ablation study, let HP-1 and HP-2 denote HosCPOOL where \mathcal{L}_{mc} in Eq. (7) has $\alpha_2 = 0$ (first-order connectivity only) and $\alpha_1 = 0$ (higher-order only), respectively.

Results are reported in Table 1. HosCPOOL achieves better performance than all baselines across most datasets. This trend is emphasised on synthetic datasets, where we know higher-order structure is critical, proving the benefits of our clustering method. DIFFPOOL often fails to converge to a good solution. MINCUTPOOL, as evoked earlier and in [41], sometimes get stuck in degenerate solutions (e.g., *Amazon PC* and *Photo* – all nodes are assigned to less than 10% of clusters), failing completely to converge even when

²SC based on motif conductance [4] instead of edge conductance; meaning SC applied on A_M .

Table 1: (Right) NMI obtained by clustering the nodes of various networks over 10 different runs. Best results are in bold, second best underlined. The number of clusters K is equal to the number of node classes. (Left) Dataset properties.

Dataset	Nodes	Edges	Feat.	K	SC	MSC	DIFFPOOL	MINCUTPOOL	HP-1	HP-2	HoscPOOL
<i>Cora</i>	2,708	5,429	1,433	7	0.150 \pm 0.002	0.056 \pm 0.014	0.308 \pm 0.023	0.391 \pm 0.028	0.435 \pm 0.032	<u>0.464</u> \pm 0.036	0.502 \pm 0.029
<i>PubMed</i>	19,717	88,651	500	3	0.183 \pm 0.002	0.002 \pm 0.000	0.098 \pm 0.006	0.214 \pm 0.066	<u>0.230</u> \pm 0.071	0.215 \pm 0.073	0.260 \pm 0.054
<i>Photo</i>	7,650	287,326	745	8	<u>0.592</u> \pm 0.008	0.451 \pm 0.011	0.171 \pm 0.004	0.086 \pm 0.014	0.495 \pm 0.068	0.513 \pm 0.083	0.598 \pm 0.101
<i>PC</i>	13,752	245,861	767	10	0.464 \pm 0.002	0.166 \pm 0.009	0.043 \pm 0.008	0.026 \pm 0.006	0.497 \pm 0.040	<u>0.499</u> \pm 0.036	0.528 \pm 0.041
<i>CS</i>	18,333	81,894	6,805	15	0.273 \pm 0.006	0.011 \pm 0.009	0.383 \pm 0.048	0.431 \pm 0.060	0.479 \pm 0.022	<u>0.701</u> \pm 0.029	0.731 \pm 0.018
<i>Karate</i>	34	156	10	2	0.792 \pm 0.035	<u>0.870</u> \pm 0.031	0.715 \pm 0.018	0.751 \pm 0.090	0.792 \pm 0.038	0.862 \pm 0.046	0.894 \pm 0.039
<i>DBLP</i>	17,716	105,734	1,639	4	0.027 \pm 0.003	0.005 \pm 0.006	0.186 \pm 0.014	0.334 \pm 0.026	<u>0.326</u> \pm 0.027	0.284 \pm 0.026	0.312 \pm 0.027
<i>Polblogs</i>	1,491	33,433	10	2	0.017 \pm 0.000	0.014 \pm 0.001	0.317 \pm 0.010	0.440 \pm 0.390	<u>0.992</u> \pm 0.003	0.994 \pm 0.001	0.994 \pm 0.005
<i>Email-eu</i>	1,005	32,770	10	42	0.485 \pm 0.030	0.382 \pm 0.019	0.096 \pm 0.034	0.253 \pm 0.028	0.317 \pm 0.026	0.488 \pm 0.025	<u>0.476</u> \pm 0.021
<i>Syn1</i>	1,000	6,243	10	3	0.000 \pm 0.000	1.000 \pm 0.000	0.035 \pm 0.000	0.043 \pm 0.008	0.041 \pm 0.006	1.000 \pm 0.000	1.000 \pm 0.000
<i>Syn2</i>	1,000	5,496	10	2	0.003 \pm 0.000	0.050 \pm 0.003	0.081 \pm 0.008	0.902 \pm 0.028	0.942 \pm 0.028	1.000 \pm 0.000	1.000 \pm 0.000
<i>Syn3</i>	500	48,205	10	5	1.000 \pm 0.000	1.000 \pm 0.000	0.067 \pm 0.001	0.052 \pm 0.002	0.115 \pm 0.006	<u>0.826</u> \pm 0.005	1.000 \pm 0.000

tuning model architecture and hyper-parameters (see Fig.2). HP-1 shows superior performance and alleviates this issue, meaning that it can be considered as an improved version of MINCUTPOOL. Spectral Clustering (SC) performs really well on some datasets, poorly on others. MSC often performs badly, revealing its excessive dependence to the presence of motifs. On the contrary, our results highlight the robustness of HoscPOOL to the limited presence of motifs due to its consideration for node features. Besides, HoscPOOL’s consideration for finer granularity levels allows to group nodes primarily based on motifs while still considering edges when necessary, which may be the reason of its superior performance with respect to HP-2, itself more desirable than HP-1 (edge-only). This ablation study proves the relevance of our underlying claims: incorporating higher-order information leads to better communities and combining several motifs further help. See Table 7 for more results.

Complexity. The main complexity of HoscPOOL lies in the derivation of A_M , which remains relatively fast for triangles: $A_M = A^2 \odot A$. In Table 2, we remark that HoscPOOL (and HP-2) has a comparable running time with respect to MINCUTPOOL on small or average size datasets. It is slower to compute than MINCUTPOOL on large datasets, while staying relatively affordable. This extra time lies with the computation and processing of the motif adjacency matrix as well as the combination of several connectivity order; which grows bigger with the graph size. Note however that we could avoid the computation of the regularisation loss, which both MINCUTPOOL and DIFFPOOL cannot afford. HP-1 is not reported as it shares similar times as MINCUTPOOL while reaching better performance.

5.2 Supervised graph classification

Experimental setup. We evaluate our pooling operator HoscPOOL on a plethora of graph classification (GC) tasks, for a fixed network architecture: GNN – Pooling – GNN – Pooling – GNN – Global Pooling – Dense ($\times 2$). Again, the GNN chosen is a GCN with skip connection, as it was found more efficient than other GNNs (see ablation study in Table 5). We sometimes add skip connections and global pooling to the output of the first and second GNN; and

Table 2: Running time (s) of the entire clustering experiment.

Dataset	DIFFPOOL	MINCUTPOOL	HP-2	HoscPOOL
<i>Cora</i>	13	16	17	24
<i>PubMed</i>	80	95	264	501
<i>Photo</i>	23	48	91	182
<i>PC</i>	89	101	304	510
<i>CS</i>	157	251	683	1406
<i>Karate</i>	9	9	9	9
<i>DBLP</i>	126	210	635	1330
<i>Polblogs</i>	8	9	10	10
<i>Email-eu</i>	9	9	10	12

concatenate the resulting vector to the third GNN’s output. Each MP layer and final dense layer has between 16 and 64 hidden units depending on the dataset regarded, and ReLU activation function. A Pooling block produces a cluster assignment matrix of dimension $num_nodes \times \text{int}(num_nodes * 0.25)$. The batch-size is different for every dataset, and ranges from 8 to 64. This architecture is trained using a learning rate for Adam of 0.001, 500 epochs, a gradient clip of 2.0, 100 early stop patience, a learning decay patience of 50 and a regularisation parameter $\mu = \{0, 0.1\}$.

Baselines. We compare our method to representative state-of-the-art graph classification baselines, involving pooling operators DIFFPOOL [49], MINCUTPOOL [5], EIGPOOL [25], SAGPOOL [19], ASAP [33], GMT [3]; by replacing the pooling layer in the above pipeline. We implement a random pooling operator (RANDOM) to assess the benefits of pooling similar nodes together, and a model with a single global pooling operator (NoPOOL) to assess how useful leveraging hierarchical information is.

Datasets. We use several common benchmark datasets for GC, taken from TUDataset [29], including three bioinformatics protein datasets *Proteins*, *Enzymes*, *D&D*; one mutagen *Mutagenicity*; one

Table 3: Graph classification accuracy. Top results are in bold, second best underlined.

Dataset	NoPOOL	RANDOM	GMT	MINCUTPOOL	DIFFPOOL	EIGPOOL	SAGPOOL	ASAP	HP-1	HP-2	HoscPOOL
<i>Proteins</i>	71.6 \pm 4.1	75.7 \pm 3.2	75.0 \pm 4.2	75.9 \pm 2.4	73.8 \pm 3.7	74.2 \pm 3.1	70.6 \pm 3.5	74.4 \pm 2.6	76.7 \pm 2.5	<u>77.0</u> \pm 3.1	77.5 \pm 2.3
<i>NCI1</i>	77.1 \pm 1.9	77.0 \pm 1.7	74.9 \pm 4.3	76.8 \pm 1.6	76.7 \pm 2.1	75.0 \pm 2.2	74.1 \pm 3.9	74.3 \pm 1.6	77.3 \pm 1.6	80.3 \pm 2.0	<u>79.9</u> \pm 1.7
<i>Mutagen.</i>	78.1 \pm 1.3	79.2 \pm 1.3	79.4 \pm 2.2	78.6 \pm 1.8	77.9 \pm 2.3	75.2 \pm 2.7	74.4 \pm 2.7	76.8 \pm 2.4	79.8 \pm 1.6	<u>81.7</u> \pm 2.1	82.3 \pm 1.3
<i>DD</i>	71.2 \pm 2.2	77.1 \pm 1.5	78.1 \pm 3.2	78.4 \pm 2.8	76.3 \pm 2.1	75.1 \pm 1.8	71.5 \pm 4.1	73.2 \pm 2.5	<u>78.8</u> \pm 2.0	<u>78.2</u> \pm 2.1	79.4 \pm 1.8
<i>Reddit-B</i>	80.1 \pm 2.6	89.3 \pm 2.6	86.7 \pm 2.6	89.0 \pm 1.4	87.3 \pm 2.4	82.8 \pm 2.1	74.7 \pm 4.5	84.1 \pm 1.1	91.2 \pm 1.0	<u>92.8</u> \pm 1.5	93.6 \pm 0.9
<i>Cox2-MD</i>	58.7 \pm 3.2	62.9 \pm 3.6	58.9 \pm 3.6	58.9 \pm 5.1	57.1 \pm 4.8	59.8 \pm 3.4	56.9 \pm 9.7	60.5 \pm 5.5	61.6 \pm 3.5	66.4 \pm 4.6	<u>64.6</u> \pm 3.9
<i>ER-MD</i>	72.2 \pm 2.9	73.0 \pm 4.5	74.3 \pm 4.5	75.5 \pm 4.0	76.8 \pm 4.8	73.1 \pm 3.8	71.7 \pm 8.2	74.5 \pm 5.9	76.2 \pm 4.2	<u>77.9</u> \pm 4.3	78.2 \pm 3.8
<i>b-hard</i>	66.5 \pm 0.5	69.1 \pm 2.1	70.1 \pm 3.4	72.6 \pm 1.5	70.7 \pm 2.0	69.1 \pm 3.1	39.6 \pm 9.6	70.5 \pm 1.7	72.4 \pm 0.8	<u>73.5</u> \pm 0.8	74.0 \pm 0.4

anticancer activity dataset *NCI1*; two chemical compound dataset *Cox-2-MD*, *ER-MD*; one social network *Reddit-Binary*. *Bench-hard* is taken from source where X and A are completely uninformative if considered alone. We split them into training set (80%), validation set (10%), and test set (10%). We adopt the accuracy metric to measure performance and average the results over 10 runs, each with a different split. We select the best model using validation set accuracy, and report the corresponding test set accuracy. For featureless graphs, we use constant features. Model hyperparameters are tuned for each dataset, but are kept fixed across all baselines. Lastly, despite being used by all baselines, note that these datasets are known to be small and noisy, leading to large errors.

Results are reported in Table 3, from which we draw the following conclusions. Performing pooling proves useful (NoPOOL) in most cases. HoscPOOL compares favourably on all datasets w.r.t. pooling baselines. Higher-order connectivity patterns are more desirable than first-order ones, and combining both is even better. It confirms findings from Section 5.1 and shows that better clustering (i.e. graph coarsening) is correlated with better classification performance. However, while the clustering performance of HoscPOOL is significantly better than baselines, the performance gap has slightly closed down on this task. Even more surprising, the benefits of existing advanced node-grouping or node-dropping methods are not considerable with respect to the RANDOM pooling baseline. Faithfully to what we announced in Section 1, we attempt to provide explanations.

5.3 Pooling behaviour investigated

First of all, we investigate the optimisation process of some key pooling operators (e.g., MINCUTPOOL, DIFFPOOL). We notice that they do not really learn to optimise their cluster assignment matrix on these graph classification tasks, producing degenerate solutions where most nodes are assigned to few clusters (similarly to Fig.2). This issue would explain why random pooling performs on par with them; as they do not learn structurally meaningful clusters.

A potential solution to this problem is to design a clustering-based pooling operator allowing to capture faithfully a more advanced kind of relationship between nodes, which we tried to do with HoscPOOL. We also tested a variety of architectures and optimisation options to see if learning would occur in specific situations. For instance, we tested several GNNs, different model architectures, skip-connections, no supervised loss at the start, etc. (see ablation

study in Table 5). However, despite clear progress – we learn to decently optimise S , to assign nodes to more clusters and to better balance the number of nodes per cluster – there still seems to be room for improvement. We thus look for other potential causes which could prevent a proper learning, especially targeting the graph classification model architecture and the nature of selected datasets.

Concerning model architecture, we show in Appendix B that using more complex clustering frameworks (*2-layer clustering*: GNN – Pooling – GNN – Pooling) prevents totally the learning of meaningful clusters for MINCUTPOOL (and DIFFPOOL), which illustrates a feature oversmoothing issue. HoscPOOL, on the other hand, has fixed this issue and still manages to learn meaningful clusters. Nevertheless, the learning process becomes longer and more difficult, leading to a drop in performance. In addition to showing the robustness of HoscPOOL with respect to existing pooling baselines, this experiment reveals that the clustering performed in graph classification tasks may not lead to meaningful clusters because of the more complex framework. Although it is likely to contribute, it is probably a factor among others, since simpler GC models like GNN – Pooling – GNN – Global Pooling – Dense (*1-pooling* in Table 5) do not improve things.

We therefore also look for answers from a dataset perspective. In Table 4, the computed graph properties and clustering results on individual graphs suggest that graphs are relatively small, with few node types co-existing in a same graph, weak homophily and a relatively poor community structure which clustering algorithms would like to exploit. Besides, because most datasets do not have dense node features (only labels), the node identifiability assumption is shaken and does not enable our MLP (5) to fully distinguish between same-label-nodes, thus making it impossible to place them in distinct clusters. On top of that, we now need to learn a clustering pattern that extends to all graphs, which is a much more complex task (compared to 1 graph in Section 5.1).

As a result, taking into consideration the multiple pooling layers, the joint optimisation with a supervised loss, the poor individual graph community structure, and the complexity of learning to cluster all graphs with few features, learning meaningful clusters becomes extremely challenging. This would explain the optimisation difficulties encountered by existing pooling operators so far. Although HoscPOOL makes a step towards better pooling, we advice future research to explore more appropriate datasets than TUDataset [29] even though it is used by all pooling baselines

Table 4: (Left) Simple graph statistics. (Middle) The clustering coefficient (cc), proportion of triangles attached per node ($triangle$), transitivity ($transi$), homophily ($homo$) and proportion of node labels in a graph w.r.t. all graphs ($diff\ labels$) are computed on each graph individually and averaged over the whole dataset. (Right) m_{sc} , sc and $sc-mod$ denote motif conductance, normalised cut, and modularity obtained by clustering each graph using traditional deterministic spectral clustering, where the number of clusters is equal to the number of labels in a graph. The last column refers the NMI obtained through HoscPool clustering only. All metrics provide information on graph community structure. *Reddit-Binary* has no node labels and is treated differently.

Datasets	# graphs	# edges	av # nodes	labels	cc	$triangle$	$transi$	$homo$	$diff-labels$	m_{sc}	sc	$sc-mod$	NMI
Proteins	1,113	162,088	39	3	.575	1.03	.517	.476	.833	.034	.005	.460	.46
NCI1	4,110	132,753	29	37	.125	.125	.214	.667	.054	.111	0.0	.388	.71
DD	1,178	843,046	284	89	.496	2.0	.462	.058	.219	.021	.013	.402	.38
Mutagenicity	4,337	133,447	30	14	.002	.003	.002	.376	.244	.056	0.0	.378	.85
Reddit-Binary	2,000	995,508	429	no	.051	.069	.009	-	-	.008	.011	.071	-
COX2-MD	303	203,084	26.2	7	1.00	103	1.00	.707	.482	.302	.333	.01	.45
ER-MD	446	209482	21.1	10	1.00	77.4	1.00	.701	.232	.331	.323	.01	.56

Table 5: Ablation study of HoscPool, denoted as *Base*. *GIN*, *SAGE*, *GAT* change the core GNN model; *No-diag* does not zero-out the diagonal of S in the pooling step, *1-pooling* uses an architecture with only one HoscPool block, *skip-co* adds a skip connection between every GNN layer and the dense layer, *c-ratio* involves a higher clustering ratio and *no-adapt* refers to the discussed dynamic adaptative loss. For *dense-feat*, we simply added some graph statistics to boost node identifiability.

Model	PROTEINS	NCI1	MUTAGEN.	DD	REDDIT-B	COX2-MD	ER-MD	B-HARD
<i>Base</i>	77.5	79.9	82.3	79.4	93.6	64.6	78.2	74.0
$\mu = 0$	76.4	78.9	80.7	78.1	93.6	64.2	75.4	73.1
<i>not-ada</i>	77.5	78.9	80.8	79.4	93.4	62.4	76.4	72.5
<i>No-diag</i>	77.7	77.2	80.0	78.9	90.2	60.9	74.6	70.7
<i>SAGE</i>	76.7	77.2	79.5	78.9	92.4	62.1	74.9	71.0
<i>GAT</i>	77.6	78.6	77.9	79.2	91.5	60.6	73.4	74.4
<i>GIN</i>	76.9	77.7	76.7	79.6	93.6	58.7	77.0	71.5
<i>skip-co</i>	77.2	77.7	80.5	79.5	93.9	61.8	76.6	71.8
<i>1-pooling</i>	76.6	79.9	82.3	78.3	90.5	63.6	77.4	74.0
<i>c-ratio</i>	75.1	77.4	80.3	78.9	92.3	61.6	75.3	70.4
<i>dense-feat</i>	77.2	79.4	80.0	78.7	92.0	58.5	73.2	70.8

as benchmark, such as Open Graph Benchmark datasets (OGB). We also recommend to design simpler node-grouping approach, to use higher-order information so as to capture more relevant communities even with complex model architectures, as well as to exploit more directly graph structure information (as targeted graphs do not have dense node features). Finally, the heterophilious nature of these datasets (Table 4) come to question the true benefit of grouping together nodes with similar embeddings (homophily assumption) when coarsening the graph.

6 CONCLUSION

We have introduced HoscPool, a new hierarchical pooling operator bringing higher-order information to the graph coarsening step, ultimately leading to motif-aware hierarchical graph representations. HoscPool builds on a novel end-to-end clustering scheme, which designs an objective function combining several continuous relaxations of motif spectral clustering, avoiding the shortcomings of deterministic methods and solving the limitations of previous key

baselines DIFFPOOL and MINCUTPOOL. The proposed experiments, through cluster observation and pooling performance, demonstrate the advantages brought by considering higher-order connectivity patterns and by combining flexibly different levels of motifs. Finally, our discussion about the relevance of the pooling operation itself aims to inspire and guide future research to design more adapted and efficient pooling operators, ensuring significant improvement over the random baseline for graph classification tasks.

Acknowledgements. Supported in part by ANR (French National Research Agency) under the JCJC project GraphIA (ANR-20-CE23-0009-01).

A SYNTHETIC DATASETS

(1) *syn1* is made of k communities, each densely intra-connected by triangles. We then widely link these communities without creating new triangles through these new links. We create random Gaussian features (included one correlated to node labels) since our method is dependent on node features.

(2) *syn2* is an Erdős–Rényi random graph with 1,000 nodes and $p = 0.012$. Each node receives label 0 if it does not belong to a triangle and label 1 otherwise. Node features include several graph statistics.

(3) *syn3* is designed using a Gaussian random partition graph with k partitions with size drawn from a normal distribution. Nodes within the same partition are connected with probability $p = 0.8$, while nodes across partitions with probability 0.2. Here, only random features are used.

B 2-LAYER CLUSTERING: PRECISIONS

In this experiment, we complexify the clustering framework (MP – MLP), making it more similar to its use as a pooling operator inside supervised graph classification tasks. More precisely, we follow an architecture: MP – Pooling – MP – Pooling. As before, the pooling step regroups an MLP to compute the first cluster assignment matrix S_1 , and a graph coarsening step. In the end, we provide a unique cluster assignment matrix S of dimension $N \times K$, composed of the two matrix derived above (S_1 and S_2), such that the probability that node i belongs to cluster k is written $S_{ik} = \sum_j S_{1ij} S_{2jk}$.

The results, given in Table 6, are obtained using 1,000 epochs with *early_stop_patience* = 500—meaning using many more epochs than for standard 1-layer clustering. This is because the convergence to a desirable solution is weaker. Furthermore, the obtained solution is less desirable and yields to a less desirable clustering. Overall, this argument is very important as it suggests that the clustering obtained in supervised graph classification tasks might not be as accurate as what our original evaluation on real-world dataset with ground-truth community structure suggested.

Dataset	MINCUTPOOL	HoscPOOL
<i>Cora</i>	0.000 \pm 0.000	0.369 \pm 0.026
<i>PubMed</i>	0.000 \pm 0.000	0.187 \pm 0.013
<i>Photo</i>	0.007 \pm 0.005	0.230 \pm 0.063
<i>PC</i>	0.004 \pm 0.001	0.194 \pm 0.042
<i>CS</i>	0.446 \pm 0.018	0.417 \pm 0.025
<i>Karate</i>	0.000 \pm 0.000	0.745 \pm 0.046
<i>DBLP</i>	0.174 \pm 0.078	0.244 \pm 0.023
<i>Polblogs</i>	0.135 \pm 0.101	0.994 \pm 0.003
<i>Email-eu</i>	0.197 \pm 0.016	0.421 \pm 0.009

Table 6: NMI of MINCUTPOOL and HoscPOOL for 2-layer clustering framework

C EXTENSION TO 4-NODES MOTIFS

Here, we consider motifs composed of 4 nodes ($|\mathcal{M}| = 4$), such as the 4-cycle or K_4 , written as $\mathbf{v} = \{l, q, r, k\}$. In Section 4.1, we formulated a relation between triangle normalised cut and graph-normalised

cut, in order to compute triangle normalised cut easily. Here, we do the same, but for 4-nodes-motif conductance. Again, we derive this relation by looking at a single cluster \mathcal{S} with corresponding

cluster assignment vector \mathbf{y} , with $y_i = \begin{cases} 1 & \text{if } i \in \mathcal{S} \\ 0 & \text{else} \end{cases}$.

$$\begin{aligned} 3\text{cut}_M^{(\mathcal{G})}(\mathcal{S}, \bar{\mathcal{S}}) &= 3 \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}\{\exists i, j \in \mathbf{v} | i \in \mathcal{S}, j \in \bar{\mathcal{S}}\} \\ &= \sum_{\mathbf{v} \in \mathcal{M}} [3(y_l + y_q + y_r + y_k) \\ &\quad - 2(y_l y_q + y_l y_r + y_l y_k + y_q y_r + y_q y_k + y_r y_k) \\ &\quad - \mathbf{1}\{\text{exactly 2 of } l, q, r, k \text{ are in } \mathcal{S}\}]. \end{aligned}$$

This expression equals $\sum_{\mathbf{v} \in \mathcal{M}} \begin{cases} 0 & \text{if all } y_l, y_q, y_r, y_k \text{ are the same} \\ 3 & \text{if 3 are the same} \\ 4 & \text{if 2 are the same.} \end{cases}$

Thus,

$$\begin{aligned} 3\text{cut}_M^{(\mathcal{G})}(\mathcal{S}, \bar{\mathcal{S}}) + \mathbf{1}\{\text{exactly 2 of } l, q, r, k \text{ are in } \mathcal{S}\} \\ &= \sum_{\mathbf{v} \in \mathcal{M}} [3(y_l + y_q + y_r + y_k) \\ &\quad - 2(y_l y_q + y_l y_r + y_l y_k + y_q y_r + y_q y_k + y_r y_k)] \\ &= \mathbf{y}^\top \mathbf{D}_M \mathbf{y} - \mathbf{y}^\top \mathbf{A}_M \mathbf{y} \\ &= \mathbf{y}^\top \mathbf{L}_M \mathbf{y} \\ &= \text{cut}^{(\mathcal{G}_M)}(\mathcal{S}, \bar{\mathcal{S}}), \end{aligned}$$

where the second inequality holds because

$$\begin{aligned} \mathbf{y}^\top \mathbf{D}_M \mathbf{y} &= \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{V}} (A_M)_{ij} = \text{vol}^{(\mathcal{G}_M)}(\mathcal{S}) = |\mathcal{M}| \text{vol}_M^{(\mathcal{G})}(\mathcal{S}) \\ &= 3 \text{vol}_M^{(\mathcal{G})}(\mathcal{S}) = 3 \sum_{\mathbf{v} \in \mathcal{M}} (y_l + y_q + y_r + y_k) \end{aligned}$$

$$\begin{aligned} \mathbf{y}^\top \mathbf{A}_M \mathbf{y} &= \sum_{i \in \mathcal{V}} y_i \sum_{j \in \mathcal{V}} y_j (A_M)_{ij} = \sum_{i, j \in \mathcal{S}} (A_M)_{ij} = \sum_{i, j \in \mathcal{S}} \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}\{i, j \in \mathbf{v}\} \\ &= \sum_{\mathbf{v} \in \mathcal{M}} 2(y_l y_q + y_l y_r + y_l y_k + y_q y_r + y_q y_k + y_r y_k). \end{aligned}$$

Overall, we obtain the following equality:

$$\text{cut}_M^{(\mathcal{G})}(\mathcal{S}, \bar{\mathcal{S}}) = \frac{1}{3} \text{cut}^{(\mathcal{G}_M)}(\mathcal{S}, \bar{\mathcal{S}}) - \frac{1}{3} \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}\{\text{exactly 2 of } l, q, r, k \in \mathcal{S}\}$$

The optimisation problem can be written as:

$$\begin{aligned} &\min_{\mathcal{S}} \sum_k \frac{\text{cut}_M^{(\mathcal{G})}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\text{vol}_M^{(\mathcal{G})}(\mathcal{S}_k)} \\ &\equiv \min_{\mathcal{S}} \sum_k \frac{\frac{1}{3} \text{cut}^{(\mathcal{G}_M)}(\mathcal{S}_k, \bar{\mathcal{S}}_k) - \frac{1}{3} \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}\{\text{exactly 2 nodes in } \mathbf{v} \in \mathcal{S}\}}{\frac{1}{3} \text{vol}^{(\mathcal{G}_M)}(\mathcal{S}_k)} \\ &\equiv \min_{\mathcal{S} \in [0,1]^{N \times K}} -\text{Tr} \left(\frac{\mathbf{S}^\top \mathbf{A}_M \mathbf{S}}{\mathbf{S}^\top \mathbf{D}_M \mathbf{S}} \right) - \sum_k \frac{\sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}\{\text{exactly 2 nodes in } \mathbf{v} \in \mathcal{S}\}}{\text{vol}^{(\mathcal{G}_M)}(\mathcal{S}_k)}. \end{aligned}$$

In practice however, unlike triangle normalised cut, this expression is not easy to compute. First of all, computing the related motif adjacency matrix is difficult; it cannot be written a simple matrix dot product. Secondly, there is this term on the RHS to take into

Table 7: Modularity (Mod), Conductance (Cond), Motif Conductance (M.Cond), Homogeneity (Homog) obtained by clustering the nodes of various networks over 10 different runs. The number of clusters K is equal to the number of node classes. HP-2 optimises better the motif conductance metric than MINCUTPOOL. HoscPool achieves a similar motif conductance but a better conductance than HP-2, which it also often outperforms in terms of modularity. Finally, MINCUTPOOL does achieve degenerate solutions for several datasets (e.g., PC, Photo, CS, Email-eu).

Dataset	MINCUTPOOL				HP-2				HoscPool			
	Mod	Cond	M.Cond	Homog	Mod	Cond	M.Cond	Homog	Mod	Cond	M.Cond	Homog
Cora	0.700	0.156	0.094	0.464	0.621	0.125	0.025	0.338	0.654	0.091	0.026	0.314
PubMed	0.532	0.120	0.047	0.225	0.478	0.069	0.029	0.101	0.454	0.082	0.038	0.096
CS	-0.005	0.001	0.000	0.000	0.684	0.141	0.087	0.637	0.695	0.131	0.084	0.638
Photo	0.000	0.008	0.002	0.002	0.566	0.084	0.033	0.470	0.684	0.093	0.043	0.580
PC	-0.001	0.000	0.000	0.000	0.546	0.285	0.263	0.457	0.591	0.149	0.082	0.556
DBLP	0.533	0.182	0.157	0.363	0.588	0.131	0.065	0.277	0.608	0.114	0.066	0.318
Karate	0.370	0.269	0.281	0.543	0.389	0.192	0.088	0.715	0.417	0.217	0.133	0.861
Email-eu	0.002	0.011	0.003	0.025	0.189	0.455	0.382	0.166	0.185	0.488	0.396	0.208
Polblogs	0.409	0.090	0.048	0.991	0.409	0.087	0.035	0.993	0.429	0.073	0.035	0.991

consideration. And although we might be able to compute both directly via a complex algorithm, it is not guaranteed that solving this problem is quicker than the original optimisation problem (def. of $\text{vol}_M^{(G)}$ and $\text{cut}_M^{(G)}$).

REFERENCES

- [1] Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. 2019. Link prediction via higher-order motif features. *arXiv preprint arXiv:1902.06679* (2019).
- [2] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*. 1993–2001.
- [3] Jinheon Baek, Minki Kang, and Sung Ju Hwang. 2021. Accurate Learning of Graph Representations with Graph Multiset Pooling. *arXiv preprint arXiv:2102.11533* (2021).
- [4] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.
- [5] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*. PMLR, 874–883.
- [6] Aldo G Carranza, Ryan A Rossi, Anup Rao, and Eunye Koh. 2020. Higher-order clustering in complex heterogeneous networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 25–35.
- [7] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.
- [8] Fan Chung. 2007. Four proofs for the Cheeger inequality and graph partition algorithms. In *Proceedings of ICCM*, Vol. 2. Citeseer, 378.
- [9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.
- [10] Frederik Diehl. 2019. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990* (2019).
- [11] Dhivya Eswaran, Srijan Kumar, and Christos Faloutsos. 2020. Higher-order label homogeneity and spreading in graphs. In *Proceedings of The Web Conference 2020*. 2493–2499.
- [12] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. 2018. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 869–877.
- [13] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *international conference on machine learning*. PMLR, 2083–2092.
- [14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.
- [15] Lun Hu, Jun Zhang, Xiangyu Pan, Hong Yan, and Zhu-Hong You. 2021. HiSCF: leveraging higher-order structures for clustering analysis in biological networks. *Bioinformatics* 37, 4 (2021), 542–550.
- [16] Rong Jin, Feng Kang, and Chris Ding. 2005. A probabilistic approach for optimizing spectral clustering. *Advances in neural information processing systems* 18 (2005), 571–578.
- [17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [18] Christine Klymko, David Gleich, and Tamara G Kolda. 2014. Using triangles to improve community detection in directed networks. *arXiv preprint arXiv:1404.5874* (2014).
- [19] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *International Conference on Machine Learning*. PMLR, 3734–3743.
- [20] John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunye Koh, and Anup Rao. 2018. Higher-order graph convolutional networks. *arXiv preprint arXiv:1809.07697* (2018).
- [21] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.
- [22] Jianxin Li, Hao Peng, Yuwei Cao, Yingdong Dou, Hekai Zhang, Philip Yu, and Lifang He. 2021. Higher-order attribute-enhancing heterogeneous graph neural networks. *IEEE Transactions on Knowledge and Data Engineering* (2021).
- [23] Ning Liu, Songlei Jian, Dongsheng Li, Yiming Zhang, Zhiqian Lai, and Hongzuo Xu. 2021. Hierarchical Adaptive Pooling by Capturing High-order Dependency for Graph Representation Learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).
- [24] Enxhell Luzhnica, Ben Day, and Pietro Lio. 2019. Clique pooling for graph classification. *arXiv preprint arXiv:1904.00374* (2019).
- [25] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. 2019. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 723–731.
- [26] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.
- [27] Diego Mesquita, Amauri H Souza, and Samuel Kaski. 2020. Rethinking pooling in graph neural networks. *arXiv preprint arXiv:2010.11418* (2020).
- [28] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.
- [29] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).
- [30] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4602–4609.
- [31] Yunsheng Pang, Yunxiang Zhao, and Dongsheng Li. 2021. Graph pooling via coarsened graph infomax. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2177–2181.
- [32] Alan Perotti, Paolo Bajardi, Francesco Bonchi, and André Panisson. 2022. GRAPHSHAP: Motif-based Explanations for Black-box Graph Classifiers. *arXiv*

- preprint arXiv:2202.08815* (2022).
- [33] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. 2020. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5470–5477.
- [34] Ryan A Rossi, Anup Rao, Sungchul Kim, Eunyee Koh, Nesreen K Ahmed, and Gang Wu. 2019. Higher-order ranking and link prediction: From closing triangles to closing higher-order motifs. *arXiv preprint arXiv:1906.05059* (2019).
- [35] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review* 1, 1 (2007), 27–64.
- [36] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. 2020. Higher-order explanations of graph neural networks via relevant walks. *arXiv preprint arXiv:2006.03589* (2020).
- [37] Govind Sharma, Aditya Challa, Paarth Gupta, and M Narasimha Murty. 2021. Higher-Order Relations Skew Link Prediction in Graphs. *arXiv preprint arXiv:2111.00271* (2021).
- [38] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.
- [39] Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3693–3702.
- [40] Konstantinos Sotiropoulos and Charalampos E Tsourakakis. 2021. Triangle-aware Spectral Sparsifiers and Community Detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1501–1509.
- [41] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2020. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904* (2020).
- [42] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*. 1451–1460.
- [43] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.
- [44] Dorothea Wagner and Frank Wagner. 1993. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*. Springer, 744–750.
- [45] Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. 2007. Trace ratio vs. ratio trace for dimensionality reduction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.
- [46] Yu Guang Wang, Ming Li, Zheng Ma, Guido Montufar, Xiaosheng Zhuang, and Yanan Fan. 2020. Haar graph pooling. In *International conference on machine learning*. PMLR, 9952–9962.
- [47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
- [48] Yuhua Xu, Junli Wang, Mingjian Guang, Chungang Yan, and Changjun Jiang. 2022. Multistructure Graph Classification Method With Attention-Based Pooling. *IEEE Transactions on Computational Social Systems* (2022).
- [49] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804* (2018).
- [50] Hualei Yu, Jinliang Yuan, Hao Cheng, Meng Cao, and Chongjun Wang. 2021. GSAPool: Gated Structure Aware Pooling for Graph Representation Learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*. 1–8. <https://doi.org/10.1109/IJCNN52387.2021.9534320>
- [51] Hao Yuan and Shuiwang Ji. 2020. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*.
- [52] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [53] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.